



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/663,455	09/15/2003	David Abu Ghazaleh	RSW920030055US1	2196
25259	7590	05/05/2009	EXAMINER	
IBM CORPORATION			VU, TUAN A	
3039 CORNWALLIS RD.				
DEPT. T81 / B503, PO BOX 12195			ART UNIT	
RESEARCH TRIANGLE PARK, NC 27709			PAPER NUMBER	
			2193	
		NOTIFICATION DATE		DELIVERY MODE
		05/05/2009		ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

RSWIPLAW@us.ibm.com

Office Action Summary	Application No.	Applicant(s)	
	10/663,455	GHAZALEH ET AL.	
	Examiner	Art Unit	
	TUAN A. VU	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1) Responsive to communication(s) filed on 03 March 2009.
 2a) This action is FINAL. 2b) This action is non-final.
 3) Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4) Claim(s) 1-22 and 25-35 is/are pending in the application.
 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
 5) Claim(s) _____ is/are allowed.
 6) Claim(s) 1-22 and 25-35 is/are rejected.
 7) Claim(s) _____ is/are objected to.
 8) Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9) The specification is objected to by the Examiner.
 10) The drawing(s) filed on _____ is/are: a) accepted or b) objected to by the Examiner.
 Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
 Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
 11) The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12) Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
 a) All b) Some * c) None of:
 1. Certified copies of the priority documents have been received.
 2. Certified copies of the priority documents have been received in Application No. _____.
 3. Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) <input type="checkbox"/> Notice of References Cited (PTO-892)	4) <input type="checkbox"/> Interview Summary (PTO-413)
2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948)	Paper No(s)/Mail Date. _____ .
3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO/SB/08)	5) <input type="checkbox"/> Notice of Informal Patent Application
Paper No(s)/Mail Date _____.	6) <input type="checkbox"/> Other: _____ .

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 3/03/09.

As indicated in Applicant's response, claims 1-3, 6-7, 12-14, 16, 20-22, 25, 27-29, 32, 33-35 have been amended, and claims 23-24 canceled. Claims 1-22, 25-35 are pending in the office action.

Claim Objections

2. Claim 1 is objected to because of the following grammatical informality: paragraph (5) contains a syntax recited as 'line between each symbol drawn in step 4 and another symbol in the graphical representation corresponding to an object (to which said object ... drawn in step 4 is assigned **OR**) within which is defined'. The underlined portion ending with 'within which IS defined' is not describing the clear relationship between any subject and its verb. The verbal clause 'within which is defined' is deemed a idiomatic phraseology and will not be given any semantic merits.

3. Claim 22 contains grammatical indefiniteness in '... line between each symbol ... and another symbol ... corresponding to an object to which it is assigned or within which it is defined' such there is no clear relationship between the verbal group 'is assigned' and 'is defined' with any *object* or *symbol* recited prior to this awkward grammatical construct. The 'it' of 'assigned' or 'defined' will be given broad interpretation.

4. Claim 16 is objected to because of the following informalities: the ";" at the end of 'at least the following five additional symbols;" is identified as a typo error, whereas it should be a colon.

Specification

5. The disclosure is objected to because of the following informalities: there is no explicit mention regarding code instructions embodied in a *computer readable media*. That is, the computer readable media recited in claim 22 is a well-known concept describing how a computer product (including program instructions) can be embodied into. However, in order determine whether a claimed product is statutory and in possession by the inventor, the claimed *product* would have to be matched against the Specifications for reasonable teaching. The Disclosure does not provide a remote mention about implementing the GUI tool or intended software program (i.e. with executable *instructions*) with a computer readable medium, signal carrier or standard storage disc. Thus, the Specifications is objected to for failing to provide description regarding the above recited ‘computer readable media’ (storing instructions code) because product claimed as readable media can be non-tangible signal, wave and would not categorize as one of the four permissible statutory subject matter. The recited product of claims 22-35 is temporarily treated as storage medium that can be read by a host computer, and pending correction to the above, claim 22 would be subjected to lack of description and/or 101 type deficiency in case no correction is made.

Appropriate correction is required.

Claim Rejections - 35 USC § 112

6. The following is a quotation of the first paragraph of 35 U.S.C. 112:

The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

7. Claims 1-22, 25-35 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the enablement requirement. The claim(s) contains subject matter which was not described

in the specification in such a way as to enable one skilled in the art to which it pertains, or with which it is most nearly connected, to make and/or use the invention.

8. Claim 1 recites steps of providing a GUI, selecting an object, drawing a main object symbol, drawing a symbol of objects assigned to the main object, drawing a line between said objects. The claim as a whole amounts to results based on actions taken or operating upon structures or objects from the *providing* step. The steps recited as *selecting* (1) and *drawing* (3, 4, 5), upon careful studying of the Specifications are construed as steps performed manually (*), i.e. by a human developer or architect (Specifications: *user can drag and drop* – 3rd para pg. 4; *preparer, architect ... by drawing* - pg. 5 bottom; pg. 7 bottom; middle and last para - pg. 8; bottom, pg. 13; *if desired ... may be drawn* – top para, pg. 15; *one may double click* – middle pg. 15; *mouse click, architect may decided to show* – middle pg. 16; bottom pg. 17; *mouse click* – middle pg. 19; *by the user* – bottom pg. 19; *double click* – middle pg. 20; *one may choose ... he or she would create* – 2nd para pg. 21; *user clicking* - bottom half pg. 21, top pg. 22). None of the *drawing* instances (e.g. ‘is drawn’ ‘may be drawn’ ‘are drawn’ ‘by first drawing’ ‘by drawing’) found in the Specifications spell out or reasonably entail an automated act of drawing achieved by underlying code in execution (*using said specially programmed computer*) independent from user-driven events and/or in total absence of any human intervention. And this manual/human aspect of ‘drawing’ is all the more evident when the graphical interface is disclosed (Summary: 3rd para pg. 4) as presentation ‘within which *the user can* drag and drop various symbols ... and *interconnect them* and fill in the object attribute’ (**). The inventor is deemed not in possession of any hardware and/or software means equipped with functions and structures in place that would clearly establish realization of the steps of *selecting* and *drawing*.

One would not be able to make use of the invention when the keys actions taken and (functional elements: selecting, drawing) operating upon the provided GUI environment (or static elements) are manually achieved by a human being, which cannot be a part integral to the invention. The above steps would be treated as though the interface thus provided operates as a container to receive results/events from user action within the interfacing environment.

Claims 2-21 for failing to clarify that the steps of selecting and drawing are not by human manual action, are also rejected for being non-enabled.

9. Claim 22 recites ‘computer readable product embodied on a computer readable media ... for enabling a user to generate a graphical representation ...’ comprising a) instructions that provide a GUI presenting symbols for use in the graphical representation, b) instructions that enable the user to select; c) instructions that enable the user to draw; d) instructions that enable the user to drag and drop; e) instructions that enable the user to draw a line; f) instructions that enable the user to denote. Scanning the Disclosure for a product in the form of computer readable media, one would only see computer program to be executed on computer; but no explicit of a form of product as claimed. More importantly, as set forth above from studying the Disclosure to see whether the selecting and drawing are automated by a software product, it has been identified that selecting and drawing, mouse clicking, dragging as recited in b) c) d) e) f) are mainly manual action by a human architect or developer --refer to (*) and (**) from above. The disclosure does not come with a description (see Objections to the Specifications) clearly showing how a software product having instructions embodied on a computer medium, while executing would provide a GUI interface as set forth in step a). Based on well-known practices, one of ordinary skill in the art would acknowledge that a computer screen would serve as a GUI

environment to receive and collect user events but would not be sufficiently taught of very details regarding how the inventor implements a software product as claimed (when submitted for being executed on a host computer) would enable the user to achieve the above step b) to f). Since, the architect or the developer can possess his/her own manual and mental capability to draw or click a mouse for drag and drop, one would not be sufficiently taught in order to make use of the invention in terms of its functional elements recited in b) to f) since human actions cannot be part of the inventor's work; nor are the (computer) instructions, even when running on the computer, deemed sufficiently described in specific details (by reading the Disclosure) as to substantially enable a human (architect or developer) actions and realize what appears to be intrinsic human manual capacities or decision making. For the sake of the prosecution, the above steps a) to f) as though the interface thus provided operates as a container to receive results/events from user action within the interfacing environment.

Claims 23, 25-35 are also rejected for failing to remedy to the lack of description as identified from above.

Claim Rejections - 35 USC § 103

10. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

11. Claims 1-2, 5-22, and 26-35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bailey (USPN: 6,701,513) in view of Kodosky (USPubN: 2003/0184580).

As per claim 1, Bailey discloses a method for graphically representing object oriented programming logic, the method comprising the steps of:

- (1) providing, via a specially programmed computer, a graphical user interface presenting a plurality of different symbols for use in a diagram of object oriented programming logic, each different symbol representing a different type of object in object oriented programming (e.g. 402, 402a fig. 4A; 430a, 440, 436a – Fig. 4D; *object members ... class methods ... object classes* - col. 7, lines 57-67; Col. 8, lines 24- 26);
- (2) selecting an object (e.g. Form 1 object 404 – Fig. 4D; col. 10 lines 36-40, 53-55) of the logic to be represented in the diagram;
- (3) drawing, using said specially programmed computer, a symbol corresponding to the selected object (e.g. object 404 Fig. 4D) and labeling the symbol with a label descriptive of the object's features so that it is distinguishable from other symbols of the same object type (e.g. col. 10, line 53 to col. 11 line 2; col. 11, lines 35-38; 426a 434a – Fig. 4D);
- (4) for each object assigned to or defined within the selected object, drawing, using said specially programmed computer, a symbol corresponding to that object and labeling the symbol with a label descriptive of the object's features (e.g. col. 10, lines 53-55, 58-59, 64-65; col. 11, lines 35-38; 430a, 426a, 434a - Fig. 4D); and
- (5) drawing, using said specially programmed computer, a line between each symbol drawn in step (4) and another object symbol in the graphical representation corresponding to an object to which said object corresponding to said symbol drawn in step (4) it is assigned or within which it is defined (e.g. col. 4, lines 23-26; link 440 – Fig. 4D; 426, 434 -Fig. 8A;

graphically linking - col. 13 lines 8-20; elements 902, 914, 026, 916, 918 - Fig. 9; col. 4 lines 29-50).

However, Bailey does not explicitly teach selecting an object in (2) as a *main object* of the logic to be represented in the diagram and drawing a symbol corresponding to the main object. Kodosky, in analogous graphical development endeavor, teaches creating an icon (e.g. paragraph [0011] lines 10- 12) representing a main program in a hierarchical view of the system (e.g. paragraph [0012] lines 9-11, 24-25). It would have been obvious to one of ordinary skill in the art at the time of the invention was made to combine the teachings of Bailey and Kodosky because Kodosky's teaching of main object with symbol would help user to easily understand logic that was described in detail according to specification of the system without creating any confusion.

As per claim 2, Bailey teaches the method, further comprising the step of: (6) providing a plurality of additional different symbols for use in the diagram, each of the additional different symbols representing a different object-oriented programming element type (e.g. col. 4, lines 23-26; *wire object class, Wire1, Wire2, wire control object* – col. 14 lines 21 to col. 15 line 49; elements 902, 914, 026, 916, 918 - Fig. 9) other than an object.

As per claims 5-6, Bailey teaches the method, wherein the labels comprise text (e.g. col. 10, lines 53-55); wherein step (5) comprises drawing the line between the symbol corresponding to the object defined in step (4) and the symbol corresponding to another object it is most directly assigned to or is most directly defined within (e.g. col. 4, lines 36-40).

As per claims 7 and 8, Kodosky teaches a method of visually creating a distributed system design and software programming which can be used in documenting software and to

prepare a program specification (e.g. see *Print documentation* – Fig. 20A). One would be motivated to enable Bailey's framework building instance so that existing building instance or pre-existed instances of build to be documented and distributed for further builds, data acquisition and testing (as suggested in Bailey: col. 3 lines 24-36)

As per claim 9, Bailey teaches the method, further comprising the step of: (8) repeating steps (1) - (5) to prepare a plurality of separate diagrams corresponding to separate parts of an overall application (e.g. col. 8, lines 14-18).

However, does not explicitly teach a first object is the main object appearing in at least a first one of the diagrams and is not a main object appearing in at least a second one of the diagrams.

Kodosky teaches a first object is the main object appearing in at least a first one of the diagrams and is not a main object appearing in at least a second one of the diagrams (e.g. paragraph [0015]). One would have been motivated to use Kodosky's main object to trigger the graphical enlistment of further objects in Bailey's endeavor, based on the connecting of icons as set forth in claim 1.

As per claim 10, Bailey does not disclose wherein the second diagram does not disclose objects assigned to and defined within the first object and the first diagram does disclose objects assigned to and defined within the first object. Based on the hierarchy of OO objects in Kodosky, and the well-known concept that base class include members defined therein in set forth in Baileys, the above submember of a base class would have flow out of the OOP approach in both Bailey (refer to claim 1) and Kodosky. Enabling a base class to disclose its defined member as opposed to the other way around (e.g. the second object not disclosing base/first

object it is assigned to) would have been a obvious feature adopted in Bailey (in light of Kodosky)

As per claim 11, Bailey teaches the method, an application-level representation disclosing an overall software system (e.g. col. 8 lines 14-18)

As per claim 12, Bailey does not explicitly disclose wherein the label for the symbol corresponding to the first object in the second diagram identifies the first diagram as disclosing further details of the first object. Bailey teaches a GUI-based object-oriented enlistment of base objects (or graphical representation thereof) prior to expanding the base with submembers (or graphical representation thereof) wherein the label identifies as disclosing further details of the object (e.g. col. 9, lines 50-53). Enabling object in graphical representation of a base class to identifies itself even when such identification is shown in a base class or submember would have been obvious; and this is evidenced with OO nomenclature where a subclass (second diagram) has a label identifying a parent class, separated by a “.” (label of a first object). Bailey teaches object-oriented building with labeling using this nomenclature (e.g. Label1.Caption, Form1.Label1 Fig. 8B), it would have been obvious for one skill in the art at the time the invention was made to implement the OOP by Bailey (in view of Kodosky) so that labeling would adopt this OO nomenclature such that the first class or object (or symbol representing it) is represented in the subclass or second object representation, i.e. the base object identifies itself in the labeling of the sub class.

As per claim 13, Bailey does not explicitly teach wherein the symbols representing different object types include at least the five following: a symbol for representing objects that are application type objects; a symbol for representing objects that are window type objects; a

symbol for representing objects that are class type objects; a symbol for representing objects that are event script type objects; and a symbol for representing objects that are method type objects. Bailey does teach plurality of objects or icons representing text boxes, radio buttons, scroll bars, menu bars and so on (e.g. col. 2, lines 3-9; col. 7, lines 57-61; col. 8, lines 8-11, 15-18, 24-37; Form.Label1, Form.Break1, Form.GreaterThan1, Form.CBAdd1, Form.TextBox - Fig. 14B-C; Form1.VScrollbar1 Form1.Label1 – Fig. 4D; col. 20 lines 32-67 – Note: icon being clicked for enabling user to specify/define event handler action OR Form1.xxx is analogized to even script type; or active icon symbols, such that a mouse event thereon reads on triggering code underlying such event); that is, each icon represents a corresponding object class that is available for use by developer (e.g. Fig. 9). It would have been obvious to one of ordinary skill in the art at the time of the invention was to add more graphical icons representing other OO programming types, i.e. symbols for application type, for window type, for class type, event script object, for method type as from above, which would improve readability of target application, container and contained elements relationship, event flows and membership of objects and corresponding meta-information or annotation specifying properties of members being developed, visibility of the OO hierarchical structuring of class members and their functional dependency with respect to a global scope application or another target subcomponent of a larger system type.

As per claim 14, Bailey does not explicitly disclose wherein the symbols representing different object types include at least the following five symbols: a first symbol for representing objects that are application type objects; a second symbol for representing objects that are window type objects; a third symbol for representing objects that are class type objects; a fourth

symbol for representing objects that are event script type objects; and a fifth symbol for representing objects that are method type objects; and

wherein the symbols representing additional program elements include:

a sixth symbol for representing data transfer; a seventh symbol for representing databases; an eighth symbol for representing remote links; and a ninth symbol for representing inheritance.

However, these symbols are similar to those in claim 13 with added symbols; therefore, it is rejected for the same rationale as claim 13; and further Bailey teach (i) data transfer type and (ii) inheritance type symbols ((i) Form.Wait1, Form.User1; Form.Yield1 – Fig. 16; (ii) Form1.Variable1; Form1.CommandButton1 – Fig. 14A). Bailey disclose remote queries of COM objects (col 24 li. 48-65), hence it would have been obvious for one skill in the art at the time the invention was made to provide symbols denoting database type and remote link type because objects retrieved from a COM paradigm (Bailey: col. 8 lines 37-43) in light of the factory of objects (Bailey: col 7 lines 61-66) would necessitate links and database identifications.

As per claim 15, Bailey teaches the method, wherein the sixth, eighth, and ninth symbols are drawn connecting two other object symbols (e.g. col. 4, lines 35-40)..

As per claim 16, Bailey does not explicitly disclose wherein the symbols representing different object types further include at least the following five additional symbols: a tenth symbol for representing objects that are menu type objects; a eleventh symbol for representing objects that are frame type objects; a twelfth symbol for representing objects that are button type objects; a thirteenth symbol for representing objects that are data structure type objects; and a fourteenth symbol for representing objects that are not one of the other object types. But based on the rationale of claim 13, and the way the elements are depicted in Bailey

interface regarding general GUI tool icons that can be used by the user for data acquisition and manipulation of symbols (see icon 402, User interface, Data Acquisition tabs - Fig. 8B, 9) the rationale as to render the provision of GUI icons to implement the above (tenth to fourteenth) symbols obvious for the same practical benefits required in a graphical development such as that of Bailey, based on the above and the benefits as set forth in claims 13 and 14.

As per claim 17, Bailey teaches the method, further comprising the step of: providing in a separate description of the logic to be performed responsive to an event script (e.g. col. 10, lines 9-12).

As per claim 18, Bailey teaches the method, wherein the symbol representing event script type objects is drawn connected to another object that directly executes the event script corresponding to the event script symbol (e.g. Form1.VScrollbar1 ↔ Form1.Label1 – Fig. 4D).

As per claim 19, Bailey discloses interconnection of objects with line representing a relationship whereby interconnected objects do not invoke the other (see Fig. 9, 14A – Note: wire reads on not invoking) but does not explicitly teach wherein the symbol representing method type objects is drawn connected to the main object of the diagram and represents that the object is available within that main object and does not represent that the main object invokes it. The main class object linked to a secondary class has been taught in Kodosky, and the rationale as to joining OO objects with lines between base class and subclasses has been addressed in claim 1.

As per claim 20, Bailey teaches wherein step (1) comprises providing a graphical user interface in which a user is presented with a pallet containing the symbols (e.g. col. 7, lines 57-61). However, Bailey does not explicitly teach wherein steps (3) and (4) comprise *dragging and*

dropping the symbols from the pallet into a work area. Analogous to Bailey's mention of well-known tools where drag and drop (e.g. Drag – col. 11 lines 36-44) are common practices (col. 2 lines 20-30) such as in Visual Basic COM technologies, Kodosky in an analogous endeavor, teaches a design tool that enables the user to drag and drop symbols from the pallet into a workspace (e.g. paragraph [0037]; [0034] lines 6-9). Therefore, it would have been obvious to one of ordinary skill in the art at the time of the invention was made to combine the teachings of object-oriented manual manipulation in Bailey's tool using this well-known practice of drag-and-drop (evidenced in Kodosky) so that user or developer can easily visualize, select and manually join, add, and/or deploy iconic objects for implementing the logic or application based on the availability of various visual GUI components in a object-based visual developing tool, just as this drag-and-drop practice was practiced for its benefits by many visual developments technologies as set forth above, at the time the invention was made.

As per claim 21, Bailey teaches wherein step (1) comprises providing a graphical user interface in which a user is presented with a pallet containing the symbols (e.g. col. 7, lines 57-61) and wherein steps (3) and (4) comprise dragging and dropping the symbols from the pallet into a work area (e.g. col. 12, lines 35-37), and wherein the labels comprise text (e.g. col. 10, lines 53-55) and further wherein at least some of the text labels can be made to appear in the graphical representation via an action taken by a user (e.g. col. 10, lines 63- 65).

As per claim 22, Bailey teaches the invention substantially as claimed including a computer readable product embodied on computer readable media readable by a computing device for enabling a user to generate a graphical representation of object oriented programming

logic (Abstract lines 10-13; col. 3, lines 24-27; col. 7, lines 57-61; refer to claim 1), the product comprising first, second, third, fourth, fifth, and sixth executable instructions that, respectively:

- (i) provide a graphical user interface in which a user is presented with a plurality of different symbols for use in developing a graphical representation of object oriented programming logic, each different symbol representing a different type of object in object oriented programming (e.g. col. 7, lines 57-61; 402, 402a fig. 4A; 430a, 440, 436a – Fig. 4D; *object members ... class methods ... object classes* - col. 7, lines 57-67; Col. 8, lines 24- 26);
- (ii) enable the user to select an one and only one object in the diagram of the logic represented in the diagram (e.g. Form 1 object 404 – Fig. 4D; col. 10 lines 36-40, 53-55);
- (iii) enable the user to draw a symbol corresponding to the object and label the symbol with a label descriptive of the object's features (e.g. col. 10, line 53 to col. 11 line 2; col. 11, lines 35-38; 430a, 426a, 434a - Fig. 4D) so that it is distinguishable from other symbols of the same object type (Note: a label for one object teaches label for a distinguishing purpose, e.g. label 1, label 2, form 1, form 2 - see Fig. 4A-D);
- (iv) enable the user to label the symbols with a label descriptive of the corresponding object's features (see above);
- (v) enable the user to draw a line between each symbol in the workspace --by the fourth computer executable instructions-- and another symbol in the workspace (e.g. col. 4, lines 35-40; col. 9, lines 60-62; e.g. col. 4, lines 23-26; link 440 – Fig. 4D; 426, 434 -Fig. 8A; *graphically linking* - col. 13 lines 8-20) corresponding to an object to which it is assigned or within which it is defined (see Fig. 4A-B; 430a, 440, 436a – Fig. 4D; col. 4 lines 29-50); and

(vi) enable the user to graphically denote the object in the diagram by drawing another symbol around the symbol corresponding to the object

Bailey does not explicitly teach selecting an object in (ii) as a *main object* of the logic to be represented in the diagram and drawing a symbol corresponding to the main object. But this *main object , draw and label* it limitations along with the *denote* main object limitation would have been obvious in light of Kodosky --e.g. paragraph [0012] lines 9-11, 24-25-- and the corresponding rationale regarding 'main object' as set forth in claim 1.

Nor does Bailey explicitly teach (iv) instructions such to enable the user to drag and drop symbols; nor does Bailey teach (iv) to draw a line between a symbol and another object such that the symbol and the another object are *dragged and dropped* into the workspace. But this 'drag-and-drop' limitation has been addressed in claim 20.

As per claim 26, Bailey teaches the method, wherein the labels are text labels (e.g. col. 10, lines 53-55).

As per claim 27, Bailey teaches the method, further comprising: computer readable seventh instructions that enable the user to prepare a plurality of the diagrams corresponding to separate parts of an overall application and further comprising computer readable instructions for enabling the user to specify relationships between individual ones of the diagrams (e.g. col. 8, lines 14-18; col. 9, lines 53-57; col. 16 line 5 to col 17 line 18; Fig. 14A-D).

As per claim 28, Bailey teaches wherein instructions of claim 28 comprise instructions that enable the user to include references associated with symbols in one diagram identifying at least one other diagram within which the object represented by that symbol also appears (e.g Fig. 14A-D and related text).

As per claim 29, Bailey (with reference to claims 27-28) does not explicitly disclose user specifying in diagrams nature of relationships between object of first diagram and second diagram wherein: (1) the second diagram discloses additional details about the object in the first diagram; (2) the second diagram shows the object in a more abstract context than the first diagram and (3) the object is the main object of the second diagram. But the object-oriented approach in Bailey (refer to claim 1) has been further evidenced with hierarchy of objects as in Kodosky whereby the OO relationships between the hierarchical object (as represented in the first and second diagrams) are selected from the group comprising of base objects and sub-objects. Based on the well-known concept of OO methodology, the limitations (1) where a subclass exposes some reference regarding its base class;(2) where the notation of a subclass shows global (non-detailed) reference of its base class nomenclature; and (3) where the base class is depicted in the diagram notation or representation of a subclass, would all have been obvious in view of the rationale of claims 9 and 10, wherein the diagram representing a derived or sub object not fully showing the internals of its parent object which it has been sub related to or derived from.

As per claim 30, Bailey does not explicitly teach wherein the symbols representing different object types include: a symbol for representing objects that are application type objects; a symbol for representing objects that are window type objects; a symbol for representing objects that are class type objects; a symbol for representing objects that are event script type objects; and a symbol for representing objects that are method type objects.

Bailey does teach plurality of objects or icons representing text boxes, radio buttons, scroll bars, menu bars and so on (e.g. col. 2, lines 3-9; col. 7, lines 57-61; col. 8, lines 8-11, 15-18, 24-37). Each icon represents a corresponding object class that is available for use by developer. It would have been obvious to one of ordinary skill in the art at the time of the invention was to add more graphical icons representing other programming objects which will improve the functionality of the system.

As per claim 31, Bailey does not explicitly teach the method, wherein the symbols representing different object types include: a first symbol for representing objects that are application type objects; a second symbol for representing objects that are window type objects; a third symbol for representing objects that are class type objects; a fourth symbol for representing objects that are event script type objects; and a fifth symbol for representing objects that are method type objects; and wherein the additional symbols representing additional program elements include: a sixth symbol for representing data transfers; a seventh symbol for representing databases; an eighth symbol for representing remote links; and a ninth symbol for representing inheritance.

Bailey does teach plurality of objects or icons representing text boxes, radio buttons, scroll bars, menu bars and so on (e.g. col. 2, lines 3-9; col. 7, lines 57-61; col. 8, lines 8-11, 15-18, 24-37). Each icon represents a corresponding object class that is available for use by developer. It would have been obvious to one of ordinary skill in the art at the time of the invention was to add more graphical icons representing other programming objects which will improve the functionality of the system.

As per claim 32, Bailey teaches the method, seventh computer executable instructions that restrict the user to using the sixth, eighth, and ninth symbols to connect two other object symbols (e.g. col. 4, lines 35-40).

As per claim 33, it is similar to claim 30 with added symbols; therefore, it is rejected for the same rationale as claim 30.

As per claim 34, Bailey teaches the method, further comprising: computer executable instructions that enable the user to providing in a separate description of the logic to be performed responsive to an event script (e.g. col. 8, line 60 to col. 9 line 36; col. 20 lines 32-67 – Note: text enabling user to specify handler or properties using a mouse click on intended object read on separate description for response to event script).

As per claim 35, Bailey teaches the method, further comprising: computer executable instructions that enable the user to insert text associated with symbols in the workspace that can be made to appear in the workspace responsive to an action taken by a user (e.g. col. 10, lines 9-12, 53-55, 58-59, 64-65; col. 34, lines 16-20).

12. Claims 3, 4, and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Bailey (US Pat. No. 6,701,513) in view of Kodosky (US PG-Pub. No. 2003/0184580) further in view of Visio 2000 Standard Edition User Guide (Published by Visio International 1999 hereinafter Visio).

As per claim 3, Kodosky teaches (e.g. paragraph [0012] lines 9-11, 24- 25) graphically denoting the symbol in the diagram corresponding to the main object so as to distinguish it from other symbols in the diagram; but combined with Bailey, does explicitly disclose drawing another symbol around the symbol corresponding to the main object.

Visio teaches drawing another symbol around the symbol for the main object (pages 15-17). Based on a visual tool wherein highlighting of an object of interest or icon from which to draw more connected associations in Bailey (see Fig. 4A-D) and drawing a square-shape outline around an object for easy focus, it would have been obvious to one of ordinary skill in the art at the time of the invention was made to combine the teachings of Kodosky (in light of Bailey) and Visio such that user or developer can easily create, distribute and/or deploy application with identifying various components in a distributed system as a main component in the system.

As per claim 4, Visio teaches the method, wherein step (7) comprises drawing a circle completely enclosing the symbol of the main object (page 18, see shape-to-shape connections).

As per claim 25, this is are the product version the subject matter in claims 3 and 4, therefore; is rejected for the same reasons used for claims 3 and 4 above.

Response to Arguments

13. Applicant's arguments filed 3/03/09 have been fully considered but they are not persuasive. Following are the Examiner's observation in regard thereto.

USC § 103 Rejection:

(A) Applicants have submitted that the line drawn as pointed to by the Office Action has nothing to do with step (5) of claim 1, which concern with joining 2 specific “OO concepts”, whereby line drawn between these as now claimed depicts assignment of one to the other (definition of one by another) as would be OO objects (Appl. Rmrks pg. 16 bottom to pg. 17 top). A line joining even two iconic representation of OO concepts remains a line unless the meaning of this line is typically defined by some underlying metadata or by some visual representation. Object-oriented model and corresponding tool does provide a way to denote

relationship between two joined objects icons. For example, Rational Rose using UML representation methodology teaches relation (e.g. member_of or instance_of) between objects by representing a line with a particular thickness, which is further added with some triangular shape or bubble at its terminal ends, or shown with some level of shading or colored texture, a level of continuity/discontinuity in conjunction with underlying metadata describing object-to-object relationship and specificity about its nature.

There is no single detail in the language of claim 1 or claim 22 (notwithstanding the grammatical impropriety as set forth in the Claim Objections) that would compel one (reading the claim) to acknowledge that the *line drawn between* a (main) object and additional object(s) is actually that which is implemented by (or borrowed from) the methodology of Rationale Rose and thus would benefit of the OO inheritance relationship which can be visually represented with a bubble or a triangle at end of the line connecting ends (member-of, instance-of). The claim language recited as ‘line between … symbol … and another symbol … corresponding to an object to which it is assigned OR within which it is defined’ has been identified as hard to construe and such language would not be sufficient to dictate that ‘assigned’ or ‘defined’ precisely integrates the type of relationship used in Rationale Rose or UML alike.

Further, the Specifications has been analyzed and deemed insufficient in establishing that the **line drawing** to interconnect main object and other objects (as in claims 1, 22) is enabled by a computer instructions or an automated means. The claim scenario in a whole recites selecting an object then adding label and interconnecting lines to this selected object with additional objects, all of which interfacing via use of a computer. All of these actions have been identified

as using manual intervention by a human (see USC 112 Rejection) hence a human form of coordination or direction is instrumental in achieving object interconnection.

The line as it is drawn as by the sequences of actions in the claim has been interpreted as though the development purpose/intention is to create a line that includes semantic relationship of the interconnected object, relationship in terms of *defining* functional aspect of one object within that of another or *assigning* a functional context of one object with that of another (refer to CLAIM OBJECTIONS). Bailey discloses objects joined by line and as the bubble/icons at the end to symbolize a form of functional relationship or semantic for the connection between the joined lines (see Wire end 430a, 436a - Fig. 9; Fig. 14a), i.e. the line ends depicted by such bubble/icon denoting the functional dependency between the objects being joined, and the underlying metadata for each such line is depicted in a Wire properties chart (see 418, 420 Fig. 4D). So, Bailey discloses lines drawn between on object (with its underlying functionality) and another (each represented by a symbol, or label), the functional relationship of the connecting line(s) being depicted as Wire icon terminals the properties thereof being shown in a separate table. The ‘assigned to’ and ‘defined in’ at best (see Claim Objections) has been interpreted as a mere functional semantic relating two functional entities whose symbols are linked as in Bailey, such that this functional relationship (e.g. Wire properties) is depicted as line ending proper terminal bubble denoting wire and its underlying properties. Bailey's interconnected objects combine to fulfill the language identified as syntactically indefinite from above.

No matter what way Bailey has been analyzed to fulfill the ‘line … drawn’ limitation, the merits of the claimed steps has to be given proper weight in order for the argument to have justifiable grounds to overturn the rejection or the cited parts of Bailey. An invention is

evaluated by what exactly is possessed by the inventor and accordingly claimed. Patentability weight or novelty of a invention cannot be founded on decision and acts provided by a human (using the invention) since user actions cannot be part of work done by the inventor (as this alleged work is recited in the method claim 1 or product claim 22), and this deficiency is analyzed in the USC 112 Rejection. Nothing in the Specifications shows that the computer used by the Inventor when executing instructions code within the provided GUI tool specifically *enables* the user (who coordinates his mental skills with devices like mouse or keyboard) to create a linking line such that this line under support of strictly software instructions actually embeds/implies (emphasis added) the semantic that one object is assigned to (*instance_of*) or defined in (*member_of*) another object being so-linked, in the likes of Rational Rose, a methodology not remotely alluded to in the claim.

The argument is deemed non persuasive.

(B) Applicants have submitted that there is absolute no grounds that Bailey mention about denoting a main object, as starting point, in order for a rationale to extend Bailey so as to use Kodosky (Appl. Rmrks pg. 19-21). The interpretation of ‘denoting’ has been explained in the argument set forth in the previous Office Action, and the current argument is not providing sufficient basis as to why ‘denoting’ as claimed MUST be understood and construed otherwise. Further, the fact of ‘denoting’ falls under the understanding that object drawn and highlighted or selected are entirely performed by a user. The issue as whether that the inventor has invented a process that automates a ‘denoting’ feature that clearly obviates user action is yet to be matched with the issue regarding how a feature not possessed by the inventor should be given weight; as most of the actions recited in claim 1 or 22 (including that of ‘denoting’) are not deemed (see

USC 112 Rejection) achieved by code instructions run by a computer. The argument regarding how a claimed feature interpreted by the Office cannot be rendered obvious demands that *prima facie* of rebut has to be clearly established, and in so doing, defeating one if not all the prongs used in the 103 Rationale. Asserting that Bailey has no main object is like asserting that the main object is a parent object being linked to a child object based on inheritance as taught in UML notation, when the claim is silent about any of this notation. A combination of teaching has to be rebutted with grounds attacking a combination of teaching, not just dissecting one reference. The inheritance among objects should be clear in how they are organized, defined and represented graphically, and one would not know whether ‘main object’ is actually ‘parent object’ until the language clearly indicates so, in which case, the modeling features (i.e. hypothetical recital of inheritance depicted via graphical representation) implemented in the claim would be considered using a obvious methodology/mechanism well-known in the OO modeling tool. The argument seems to have been founded on the presumption that ‘denoting’ a *main object* is novel feature worth obtaining patentable merits, but this feature is argued as falling under a non-enablement type of deficiency (refer to USC 112 Rejection). The argument would be better addressed until the USC 112 Rejection is overcome, to obviate extraneous resources as any counter argument by the Office would be viewed as addressing a ‘denoting’ action (e.g. using a mouse click) done by a human being, not work done by the inventor. The argument remains unconvincing.

(C) Applicants have submitted that the claim as now amended clearly teach ‘denoting’ as making a object so as to distinguish it from other (Appl. Rmrks pg. 22). The mere act of denoting any object entails making it stand out as opposed to having it undenoted; and using a

circle or adding another symbol around it has been taught in Visio. The argument is not deemed sufficient in defeating the prongs combining Bailey, Kodosky, in light of the well known denoting features in Visio. Writing a circle or symbol around another again falls under the ambit of manual drawing not possessed by the code execution recited in claim 1 and 22 since the USC 112 Rejection remains outstanding. The argument would be better addressed until the USC 112 Rejection is overcome, to obviate extraneous resources as any counter argument by the Office would be viewed as only addressing a ‘denoting’ limitation (e.g. using a mouse click adding a contour) not possessed by the invention.

(D) Applicants have submitted that documenting ‘preexisting software’ has not been addressed properly (Appl. Rmrks pg. 24). Software being designed, modeled, implemented and tested would be followed by documentation whereby all theoretical/practical learning or results can be stored in a format for future reference. Claim 7 does not provide how work done by the inventor has been implemented in sufficient details showing how a human act which record/generate drawings or model construct and at the same time redirect these user events in generating some textual format to improve a existing document retrieved from a repository. Documentation of model construct or software design whether this document applies to present or past builds would have been an obvious feature, especially when Bailey teaches a distributed software development. Burden is on the inventor to clarify how the action of generating objects by a human (refer to claim 1) is enabled in the Disclosure so that based on said actions documenting of pre-existing documentation can be achieved. The argument would be better addressed until the USC 112 Rejection is overcome, to obviate extraneous resources because the

‘documenting’ limitation (e.g. using actions of claim 1) is deemed not possessed by the invention.

(E) Applicants have submitted that the full actual scope of claim 10 has not been addressed as the Office purports that a base class can be used to define its member (Appl. Rmrks pg. 25). The argument or rather pleading without showing of facts is hard to understand and the rejection will stand, notwithstanding the fact that all objects creation in the claimed modeling scenario are not work done by a stand-alone automated software code execution.

(F) Applicants have submitted that claims 13, 14, 16 have been amended (Appl. Rmrks pg. 27-28). The argument over a currently amended language is considered moot because the rejection has also been altered to accommodate changes in the claims.

(G) Applicants have submitted that (for claim 19) Kodosky teaches away from the ‘not invoking’ by the main object (Appl. Rmrks pg. 30). The argument is largely moot in view of the current rejection.

(H) Applicants have submitted that as amended claims 1, 22 are distinguishable over the prior art (Appl. Rmrks pg. 31). The work done by the inventor is perceived as providing a GUI tool supported by a computer, implementation thereof having no readable software instructions being stored as any product disc (see Specifications Objection); and what is apparently achieved from the *method* or *product* claim amounts to actions by a user operating on the provided interface (see USC 112 Rejection). That is, the Specifications cannot substantiate in details how ‘enable the user to’ is done or supported by instruction code in execution. As a whole, the recited actions constitute the steps by which the inventor/applicants believes/believe to have implemented or carried out (reduced to Practice) the invention. These actions have been (at length) analyzed as

being solely under direct action/intervention and mental control of a human as set forth above. It is noted that human action is unpredictable and otherwise subjected to desires, preferences and design choice of such human individual, and none of that would characterize as an automated process (possessed by the inventor using HW and SW combination) by which a result is obtained, i.e. an automated process which enables in a clear manner how the disclosed actions are accomplished. The claimed actions (e.g. joining two objects) are not recited as done in tight conjunction with information read from a algorithmic structure or a directives file. In claim 22 for example, it is not clear how instructions (embodied in a readable medium) when executed provide an automatic (purely engine driven) flow of sequence as to "enable" a *line* to go from one *main object* to another *object* should no user action be implicated as a trigger or control. As for the Specifications, there is no single paragraph that put forth or remotely teaches a clear paradigm to the effect that user action is only needed as far as to initiate, click/invoke a triggering event so that the **actual drawing**, (or labeling, denoting, drawing a object around a main object) would be automated (by software) after that event. Design choice and human actions based on such choice cannot constitute sufficient a "patentable" teaching for one to consider imparting resources towards determining how much merit can be candidate for patentability; and the deficiency in claiming only human actions (see USC 112 Rejection) would be subjected to a non-statutory type if no resolution is made.

The claims in all stand rejected as set forth in the Office Action.

Conclusion

14. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

May 01, 2009